

EE49 Project: Part 1
Communication Protocol Design & Specification
Due Date: 5/24/2011

The first part of the course project will be an entire-class exercise in protocol design. There will be no labview code due, but there will be a **class discussion on 5/24** and you will be graded based upon your participation in the discussion. This lab readme begins with a primer on the OSI model for communications, states some basic protocol requirements, and then poses a number of questions you should be ready to discuss in class. We will ask that one or two students volunteer to take notes during the discussion and write up a formal protocol specification.

GOAL: You've spent the last 4 labs learning about various aspects of communications. The aim of the course project is to bring it all together to design and implement your very own, custom communication protocol. Part 1 of this lab will require you, as a class, to create a protocol specification which everyone agrees upon. Part 2 will be implementing it!

1 Primer: Communication Protocols

(1) The OSI Model

We begin with a brief review of the OSI model for communications. We've covered some of the model layers in detail throughout this course. This contains more detail than you will need for the lab, but is provided as an FYI.

In the late 1970s, as data communications became more widespread, designers recognized the need for a standardized development approach. The International Standards Organization kicked off the creation of such a standard in 1979, calling it the the *Open Systems Interconnection* (OSI) model. The basic idea was to provide a common framework in which engineers could create interoperable communication protocols. By the late 1980s, the OSI Model had evolved into a widely adopted approach to communications system design.

The model is comprised of a set of "layers," each of which are responsible for specific functionalities of a communication system. The model also specifies the allowable interactions, or procedure calls, between every set of adjacent layers. Interaction between non-adjacent layers is never allowed. There are seven layers in total, some of which are ignored in many designs.

Figure 1 illustrates the OSI seven layer protocol stack for two users and a data switch. We provide a brief review of the OSI layers, top to bottom, which will be useful for

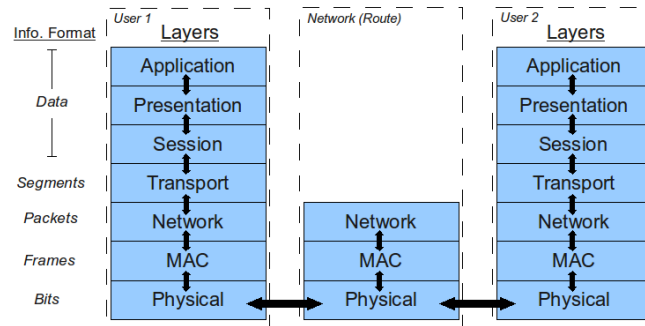


Figure 1: The OSI Reference Model

subsequent discussion:

7. **Application:** Handles application services, and as such is responsible for direct interaction with the user-defined applications. The eventual interpreter of all data transmitted/received.
6. **Presentation:** Converts between application data format and network data format, as well as handling data encryption.
5. **Session:** Opens, manages, and closes connections with other users (through interaction with their Session layer).
4. **Transport:** Fragments data segments into network packets, and handles the error-free transmission and reception of these packets.
3. **Network:** Handles the conversion between the logical destination address, as specified by the application, and the physical destination address. Additionally, this layer makes routing decisions for the packets.
2. **MAC:** Responsible for data flow over a network. Handles *channel access*, i.e. when to send information over the network, as well as converts and buffers packets into data frames. Frame-level error prevention, detection, and correction is handled at this layer.
1. **Physical:** Handles all aspects of the physical communication of a bitstream. Data en/decoding, de/modulation, and physical transmission/reception occurs at the physical layer.

The OSI standard has led to efficient protocol design because of the strict, clear boundaries between the layers. Each layer makes no assumptions about the adjacent layers, other than that they will be capable of pre-defined interactions as specified by the OSI standard. Thus, theoretically, a single layer can be upgraded or replaced

without affecting the design of the other layers. Over the past twenty years, this approach has produced modular and robust communication systems (most notably, everything Internet, which is why this approach became so popular).

Thus far, your labs have focused on layers 1 (de/modulation), 2 (error checking), and 7 (reading/writing ASCII messages) of the OSI model. In this lab, we'll pull these efforts together, as well as including more aspects of layer 2 and a simplified layer 3.

(2) Protocol Specification

Most of the protocols we use today (WiFi, etc.) follow an OSI-like design approach. The newest protocols have begun to deviate from the stricter rules of the model because of the unique nature of wireless communication, but nonetheless the OSI model remains an important paradigm in communications system design. As such, we'll use this approach for our project efforts. You will develop a protocol which spans layers 1,2,3 and 7.

To develop a protocol specification like 802.11, it takes many years and the efforts of many people from many technology vendors. Among many challenges, they often have different goals and politics in mind which lead to long discussions (and sometimes, arguments!). In the end, it's well worth the trouble though - you never have to worry about whether or not different wifi products will work together!

You will aim to implement a point-to-point wireless communication link. For extra credit, you can implement features which allow for 'multiple users'. Essentially, this amounts to creating a more complex state machine with a few more features.

We'll ask you to think about a variety of protocol communication aspects over the next few days, and have a protocol specification meeting on Tuesday, 5/24 in class. As aforementioned, we'd like one or two students to take notes and be responsible for writing up the spec document.

2 Protocol Requirements

While the design of this point-to-point protocol is up to you, we do have a few requirements that the protocol must satisfy. As you go through the questions below, think about your what you'll need to implement to meet these requirements:

1. You must implement some kind of channel coding.
2. You must implement an ACKing mechanism.

3. You must implement some form of CSMA.
4. You must know who messages are from, and who they are intended for (think: addressing).
5. Your transceiver must work in real time.

3 Protocol Design Considerations

We'll meet in class on 5/24 to hash out the details of a communications protocol which each of you will then implement. Between now and then, we ask you to consider a number of questions (by layer) so you're ready to discuss them in class on Tuesday. Before we get started, here's a few thoughts to consider:

1. You only have 2 weeks to implement the protocol, so don't overcomplicate things.
2. Plan on working in teams to get the transceivers running. You'll need to submit your own code, but you should work in teams of 2 or 3 to debug and test.

And now, for some design questions from the top layer to the bottom:

(1) Application Layer

Your requirements for the application layer will dictate many of the necessary features of the lower layers, so choose carefully!

- What are you going to send? Pictures? Text messages? (Hint: Pick the latter)
- How long will the messages be? Will you limit the length? Will you specify the length to the lower layers?
- Are your messages secret, or is it ok if everyone hears them?

(2) Network Layer

- How will you address messages? I.e, how do you tell everyone a message is from you, and for your friend?
- How long will your addresses be?

(3) Media Access Control (MAC) Layer

- How will you decide whether or not it is OK to access the medium? If it's not OK, how long will you wait to try again?
- How will you know the length of a packet you are receiving? Will you send fixed length packets or variable length?
- Will you provide error detection at this layer? What will you do if you detect an error?
- When you send a message, how long will you wait for ACKnowledgement? If you don't receive an ACK, what should you do?

(4) Physical Layer

- What signaling methodology will you use? Will you use one mapping (i.e., just BPSK) or more than one (i.e. BPSK and QPSK)? If you use more than one mapping, how will you decide which to use?
- What will you use to synchronize to the start of a packet?
- Will you use channel coding? What kind of coding will you use?
- The data which is finally presented for transmission might have a DC component, i.e. there are more '1's than '0's, or vice versa. This is not good for a variety of reasons, including saturation of the transmitting amplifier. How should we deal with this?

This is certainly not a complete list of everything to think about! If you think of other questions or concerns, come prepared to discuss them during class! See you Tuesday!